# A SEMI-SUPERVISED HEAT KERNEL PAGERANK MBO ALGORITHM FOR DATA CLASSIFICATION *

EKATERINA MERKURJEV †, ANDREA L. BERTOZZI ‡, AND FAN CHUNG §

**Abstract.** We present a very efficient semi-supervised graph-based algorithm for classification of high-dimensional data that is motivated by the MBO method of Garcia-Cardona (2014) and derived using the similarity graph. Our procedure is an elegant combination of heat kernel pagerank and the MBO method applied to study semi-supervised problems. The timing of our algorithm is highly dependent on how quickly the pagerank can be computed; we use two different yet very efficient approaches to calculate the pagerank, one of which proceeds by simulating random walks of bounded length. Overall, our method is advantageous for very big, sparse data, in which the graph has few edges, and it produces good accuracy even if the number of labeled instances is very small. In fact, the accuracy of the procedure is comparable with or better than that of state-of-the-art methods and is demonstrated on benchmark data sets. In addition to experimental results, we include a thorough comparison of our algorithm to that of Garcia-Cardona (2014) and describe the advantages of both methods.

**Key words.** heat kernel pagerank, graphs, graph Laplacian, threshold dynamics, random walk

**AMS subject classifications.**

## 1. Introduction

Classification is an important problem in machine learning and computer vision. The goal is to partition a data set efficiently and accurately into a number of clusters. This task occurs in numerous applications, such as medical or genomics data, spam detection, face recognition, financial predictions, etc. and is finding growing use in text mining studies. In this paper, we present an efficient algorithm for classification of high-dimensional data formulated in a graphical framework and motivated by [31]. The authors of [31] introduce a multi-class MBO scheme on graphs, building upon the graph Ginzburg-Landau functional that is provably close to the graph cut measure of a partition [4], and equivalent to minimizing the graph total variation functional of the classification function. Total variation has had a major impact in low-dimensional imaging processing in Euclidean space [10–12, 60] and its graphical cousin has been studied in machine learning and network science through quantities such as the Cheeger cut, Ratio cut, and modularity optimization [40, 52].

We approach the classification problem via the use of heat kernel pagerank, one of the many versions of pagerank that exist [34]. The original pagerank algorithm [58], developed in the 1990's, was designed by Google to rank the importance of websites in search engine results. However, partly due to its generality and simplicity, it was later applied to many other tasks in a variety of fields, such as biology and chemistry [34]. Overall, two particular uses comprise the majority of the applications of pagerank. First, pagerank measures the "significance" of each node in relation to the whole graph (i.e. in [45]); in other words, pagerank produces a ranking of the vertices. Second, it is often used for local graph clustering to find a cluster around a particular point (i.e. in [21]).

While most of its uses consist of ranking and local clustering, personalized pagerank,

*This work was supported by NSF grants DMS-1417674 and DMS-1118971, ONR grant N00014-16-1-2119 and AFSOR FA9550-09-1-0090. The first author was supported by the UC President's Postdoctoral Fellowship.

†University of California, San Diego, emerkurjev@ucsd.edu

‡University of California, Los Angeles, bertozzi@math.ucla.edu

§University of California, San Diego, fan@ucsd.edu

which is another version of pagerank, has been applied in a few papers to both unsupervised and semi-supervised learning. For example, in [66], the authors present a top-down algorithm based on personalized pagerank that combines random walks and modularity to accurately cluster the graph into two parts. In addition, in [23], Chung et al. formulate a procedure that uses personalized pagerank vectors and an optimized jumping parameter to obtain a set of clusters. An example of an application of the pagerank to semi-supervised learning is presented in [70], where a closed-form expression for the class of each node is derived. Moreover, the authors of [50] describe a semi-supervised method for classifying data using cleverly-designed random walks. While these algorithms produre good results, [66] is only a binary clustering procedure, [23] requires the computation of a different pagerank for every node and [70] involves solving a very large matrix system. We now present a simple, efficient and accurate algorithm for semi-supervised learning based on heat kernel pagerank.

When formulating the new method, we use heat kernel pagerank, which provides several advantages and, to the best of our knowledge, has never been applied to the semi-supervised or unsupervised machine learning task. One advantage of using the pagerank is the fact that it can be computed very efficiently, i.e. by the procedure described in [21, 62], which consists of simulating random walks of bounded length. Second, as experiments of [43] indicate, heat kernel pagerank is able to capture the properties of real-world communities better than personalized pagerank, and it is more accurate at detecting communities. Third, heat kernel pagerank represents an exponential sum, which converges more quickly in most cases than the geometric sum of personalized pagerank. In addition, using the pagerank eliminates the need to compute the eigenvalues and eigenvectors of the graph Laplacian, which are calculations that may be computationally expensive, especially in the case of very large graphs. These calculations are present in [31]. However, the method in [31] does have its own advantages, and a comparison of this algorithm and the one proposed here is given in Section 5. Overall, our approach is particularly useful in the case of large, sparse graphs.

Using heat kernel pagerank involves embedding the data in a graphical setting, which provides several advantages. First, it allows procedures to exploit non-local information and take into account the relationship between any two nodes in the data set. In the case of image processing, this makes it easier for one to capture repetitive structure and texture, as shown in [54]. The graphical setting is also very general, and can be easily applied to any data set, i.e. video data, set of images, hyperspectral data, medical data, text data, etc. Moreover, the framework provides a way to analyze data whose different clusters are *not* linearly separable.

The method we formulate is semi-supervised, meaning that a set of known labelings, called the fidelity set, is to be provided a priori. One advantage of our proposed algorithm is that, even with a very small fidelity set consisting of labeled points, usually only about 2.5%-10% of the data, it is able to obtain an accurate classification. In comparison, supervised methods usually use about 60% and 40%, or 70% and 30%, of the data for training and testing sets, respectively.

### 1.1. Previous Work

The problem of classification involves exploiting the similarities and the structure of the data to properly cluster the set into several groups. Our procedure uses the graphical framework, involving a set of vertices and edges with weights assigned on the edges, as a simple and clear way to obtain the underlying similarities in a data set.

The setting has been often used in the literature for that purpose: for example, [2, 14, 68, 71–73] consider graphs for their methods. The graphical framework has

been especially useful in image processing applications [5, 25, 26, 28, 35–37, 49, 61]. Specific applications include the image denoising method of Buades [8] and that of Elmoataz [28], which incorporates generalizations of the graph Laplacian for the task of manifold smoothing and image denoising. In addition to image processing, methods involving spectral graph theory [17, 56], based on a graphical setting, are often formulated to solve clustering problems.

More recently, the graphical setting has been used as a framework for some optimization tasks. In particular, a popular general minimization problem considered for machine learning and image processing is

$$\min_u \Big\{ E(u) = R(u) + F(u) \Big\},$$

where $R(u)$ is the regularization functional, $F(u)$ is the fidelity term and $u$ is a function defined on the space of the data set that describes an appropriate characteristic depending on the problem. The $R(u)$ term in some sense smoothes the data, and the fidelity term allows one to incorporate any known information, such as any nodes that have a known class. With regards to the regularization term, the total variation functional has been used very successfully in many image processing tasks, as shown in [13, 33, 69]. Due to some unfavorable properties of the functional, the methods in [31, 40, 54] use the Ginzburg-Landau functional as a smooth but arbitrarily close approximation to the total variation semi-norm. Our method is motivated by these algorithms, in particular, the procedure described in [31], but it uses heat kernel pagerank to overcome some computational hindarances of [31].

Heat kernel pagerank, described in greater detail in [18], has been applied in several recent works to a number of tasks. For example, [19] presents a local graph partitiong algorithm using heat kernel pagerank, which, for a subset $S$ with conductance $h$, finds local cuts with Cheeger ratio at most $O(\sqrt{h})$. In [20], the authors descibe an algorithm solving linear systems with boundary conditions using heat kernel pagerank. The method in [21] is another local clustering algorithm, which uses a novel way of computing the pagerank very efficiently. An interesting application to heat kernel pagerank is presented in [62], which outlines a method for finding a consensus value in multi-agent networks. In addition, [22] considers the problem of computing a local cluster in a massive graph in the distributed setting using heat kernel pagerank.

The classification algorithm presented in the paper is semi-supervised, so that a small set of labeled data points is provided to the method in advance. Such problems have been studied in the context of diffuse interfaces; for example, [3] introduces a procedure for image processing and machine learning in which the Ginzburg-Landau functional with fidelity term is minimized using convex splitting. Other semi-supervised learning approaches involving the functional include [31, 32, 53–55]. Semi-supervised approaches derived using total variation include [1, 51]. The advantage of our method is that, in practice, only a small percentage of ground truth needs to be inputted into the algorithm to obtain good accuracy. In fact, in our experiments, we mostly use only around 5% or less of the points as fidelity.

In the case of unsupervised methods, there is no a priori information about the labeling of the nodes of the data set. The method proceeds by clustering similar nodes together. To prevent trivial solutions, a constraint on the size of the clusters is usually incorporated into the procedure. Two such constraints that have become popular are the normalized cut [52, 67] and the Cheeger cut [15, 52, 61]; each of them favors solutions where the clusters are of similar size. The latter constraint has been studied recently in works such as [6, 7, 38, 64].

### 1.2. Main Results

The main contributions of the paper are the following:

- We introduce a very efficient new graph-based semi-supervised classification algorithm, called heat kernel pagerank MBO, for high-dimensional data that uses heat kernel pagerank for the main steps. The running time of the method depends heavily on how fast the heat kernel pagerank can be computed. We use two very efficient approaches for this task; one of them is formulated in [20–22, 62]. Our algorithm is especially useful for large, sparse graphs, and produces accuracy comparable with or better than that of state-of-the-art, even when the number of labeled instances is very small.
- We validate our method by performing experiments using benchmark data sets. We also provide a thorough comparison of this algorithm and [31], one of the state-of-the-art graph methods, and provide details on the advantages of both.

The paper is organized as follows: Section 2 provides some background on the graphical framework and heat kernel pagerank and Section 3 presents a model using heat kernel pagerank directly as a classifier. Section 4 formulates the new algorithm as well as provides details on computing heat kernel pagerank, Section 5 presents experimental results on benchmark data sets, and Section 6 concludes the paper and also provides an analysis on our method and that of [31].

## 2. Background

### 2.1. Graph Framework

In this paper, we consider a graphical framework with an undirected graph $G = (V,E)$, where $V$ is the set of vertices and $E$ is the set of edges. A weight function is defined on each edge, and represents a measure of similarity between the vertices it is connecting. As usual, the degree of a vertex $x$ is formulated as

$$d(x) = \sum_{y \in V} w(x,y).$$

Denote by $D$ the diagonal matrix containing the degrees of vertices as diagonal entries, and let $W$ denote the similarity matrix containing the weight function values. Here are some common mathematical operators on graphs (see [3, 36] for more information). Let the set of vertices $V$ and edges $E$ be Hilbert spaces defined via the following inner products and norms:

$$\langle u,\gamma\rangle_V = \sum_x u(x)\gamma(x)d(x)^r,$$

$$\langle \psi,\phi\rangle_{\mathcal{E}} = \frac{1}{2}\sum_{x,y} \psi(x,y)\phi(x,y)w(x,y)^{2q-1},$$

$$\|u\|_{\mathcal{V}} = \sqrt{\langle u,u\rangle_{\mathcal{V}}} = \sqrt{\sum_x ,u(x)^2 d(x)^r},$$

$$\|\phi\|_{\mathcal{E}} = \sqrt{\langle \phi,\phi\rangle_{\mathcal{E}}} = \sqrt{\frac{1}{2}\sum_{x,y} \phi(x,y)^2 w(x,y)^{2q-1}},$$

$$\|\phi\|_{\mathcal{E},\infty} = \max_{x,y} |\phi(x,y)|,$$

for some $r \in [0,1]$, $q \in [\frac{1}{2},1]$, $u,\gamma \in V$, and $\psi,\phi \in E$. The gradient operator is defined as

$$(\nabla u)_w(x,y) = w(x,y)^{1-q}(u(y)-u(x)),$$

and the divergence operator can be formulated as the adjoint of the gradient

$$(\mathrm{div}_w \phi)(x) = \frac{1}{2d(x)^r} \sum_y w(x,y)^q (\phi(x,y)-\phi(y,x)),$$

where the following adjoint equation is used: $\langle \nabla u, \phi \rangle_{\mathcal{E}} = -\langle u, \mathrm{div}_w \phi \rangle_{\mathcal{V}}$. Using the gradient and divergence operators, one can define a family of graph Laplacians $\triangle_r = \mathrm{div}_w \dot{\nabla} : \mathcal{V} \to \mathcal{V}$:

$$(\triangle_w u)(x) = \sum_y \frac{w(x,y)}{d(x)^r}(u(y)-u(x)).$$

We also formulate the Dirichlet energy and total variation:

$$\frac{1}{2}\|\nabla u\|_{\mathcal{E}}^2 = \frac{1}{4}\sum_{x,y} w(x,y)(u(x)-u(y))^2,$$

$$TV_w(u) = \max\left\{\langle \mathrm{div}_w \phi, u\rangle_{\mathcal{V}} : \phi \in \mathcal{E}, \|\phi\|_{\mathcal{E},\infty} \le 1\right\} = \frac{1}{2}\sum_{x.y} w(x,y)^q |u(x)-u(y)|.$$

If $u$ is viewed as a vector in $\mathbb{R}^m$, we can write

$$-\triangle_w u = (D^{1-r} - D^{-r}W)u.$$

The expression $D^{1-r} - D^{-r}W$ forms a general expression for the graph Laplacian, which is the graph version of the Laplace operator. The parameter $r$ allows the user to control the type of Laplacian. For example, the cases of $r=0$ and $r=1$ result in the unnormalized Laplacian $L = D - W$ and the random walk Laplacian $L_w = D^{-1}L$, respectively. Another popular version of the Laplacian is the symmetric Laplacian $L_s = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

These graph Laplacians have the following easily shown properties:
1) $L$ and $L_s$ are symmetric.
2) $L$, $L_s$ and $L_w$ are positive semi-definite.
3) $L$, $L_s$ and $L_w$ have non-negative, real-valued eigenvalues $0 = \lambda_1 \le \lambda_2 \le ... \le \lambda_n$.
4) $\lambda$ is an eigenvalue of $L_w$ with eigenvector $u$ if and only if $\lambda$ is an eigenvalue of $L_s$ with eigenvector $w = D^{\frac{1}{2}} u$.

With regards to the weight function $w$, the goal is to choose $w$ so that similar vertices are weighted by a large value and dissimilar vertices are weighted by a small value. While many versions of weight functions exist, two popular ones are the Gaussian weight function

$$w(x,y) = e^{-\frac{d(x,y)^2}{\sigma^2}}$$

and the Zelnik-Manor and Perona weight function [59], which gives a local rescaling, and is written as

$$w(x,y) = e^{-\frac{d(x,y)^2}{\sqrt{\tau(x)\tau(y)}}},$$

where $d(x,y)$ is a metric computing the distance, in some sense, between $x$ and $y$, and $\sigma$ is a parameter to be chosen. Also, $\sqrt{\tau(x)} = d(x,z)$ and $z$ is the $M^{th}$ closest vertex to $x$, where $M$ is a variable to be chosen.

The metric $d(x,y)$ will depend on the data set. For example, in the case of points in $R^2$, one might consider it to be the usual Euclidian distance, since points far apart are more likely to be in separate clusters than those closer together. When working with images, one might consider comparing neighborhoods around pixels $x$ and $y$ using the $L^2$ or $L^1$ norm. Of course, techniques like rotation invariance can also be incorporated. For text classification, one can use $tfidf$ (term frequency inverse document frequency) to form feature vectors for each document and then use cosine similarity to compare the vectors. In the case of hyperspectral data, one way to formulate the metric $d(x,y)$ would be to incorporate the hundreds of channels of each node.

### 2.2. Heat Kernel PageRank and Personalized PageRank

The heat equation, formulated as

$$\frac{\partial u}{\partial t} = \Delta u,$$

occurs very frequently in many different kinds of applications. For example, in financial mathematics, it is used to solve the Black-Scholes partial differential equation. A more general version of it, called the diffusion equation, occurs frequently in chemical diffusion steps. In this section, we show some ways of solving the heat equation in a graphical setting:

$$\frac{\partial u}{\partial t} = -Lu, \tag{2.1}$$

where the Laplace operator $\Delta$ is replaced by the graph Laplacian. Different Laplacians, such as $L_s$ or $L_w$, can be used.

The *heat kernel* of a graph $G$ is the matrix $e^{-tL}$, which is a solution to the heat equation (2.1). If the symmetric graph Laplacian is used, the heat kernel is $\mathcal{H}_t = e^{-tL_s}$. This version is of interest due to the fact that $L_s$ is a symmetric matrix. If the random walk Laplacian $L_w = I - D^{-1}W = I - P$ is considered, the resulting heat kernel is $H_t = e^{-t(I-P)}$. Note that $P = D^{-1}W$ is just a transition probability matrix for the random walk in which one moves from vertex $i$ to vertex $j$ with probability given by $(D^{-1}W)_{ij}$.

The *heat kernel pagerank* is defined as $\rho_{t,f} = fH_t$ for row vector $f$:

$$\rho_{t,f} = fH_t = \sum_{k=0}^{\infty} e^{-t}\frac{t^k}{k!} fP^k. \tag{2.2}$$

From (2.2), one can see that, in the case that $f$ is a probability distribution, the heat kernel pagerank is just the expected distribution of the random walk with probability matrix $P$, where $k$ steps are taken with probability $e^{-t}\frac{t^k}{k!}$, and the starting point is chosen from $f$. Therefore, in this case, one can approximate the heat kernel pagerank vector by simulating enough random walks (with the number of steps calculated according to the Poisson distribution, and with the starting point taken from $f$) to obtain a close approximation to the expected distribution. This approach is the one taken by [21, 62] when computing heat kernel pagerank.

In [62], Simpson and Chung show a useful application of heat kernel pagerank. Consider the following heat equation in a graphical setting:

$$\dot{u}(t) = -L_w x(t), \qquad u(0) \in R^n. \tag{2.3}$$

According to Theorem III.2 in [62], the solution to (2.3) is given by

$$u(t) = D^{-1}\rho_{t,f}^{tr}, \qquad f = u(0)^{tr}D, \tag{2.4}$$

where $M^{tr}$ denotes the transpose. Therefore, the heat equation in a graphical setting, defined using the random walk Laplacian can be solved using heat kernel pagerank. Other applications of heat kernel pagerank include finding the consensus value in multi-agent networks [62], local clustering [19, 21, 22] and solving local linear systems with boundary conditions [20].

Another version of pagerank is *personalized pagerank* $pr_{\alpha,f}$. It is defined by the following recursion formula:

$$pr_{\alpha,f} = \alpha f + (1-\alpha)pr_{\alpha,f}P, \tag{2.5}$$

where $P$ is a transition probability matrix, $f$ is a probability distribution and $\alpha$ is a scalar between 0 and 1. The pagerank describes a surfer model in which a web surfer, with probability $\alpha$, jumps to a page chosen from a probability distribution $f$, but with probability $1-\alpha$, moves according to the transition probability matrix. It also represents the probability distribution of the random walk arriving at each node. An equivalent definition of the pagerank is:

$$pr_{\alpha,f} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k f P^k. \tag{2.6}$$

Comparing the two different pageranks defined by (2.6) and (2.2), we see that the former is a geometric sum and the latter is an exponential sum. Very often, the exponential sum converges more quickly.

### 3. Heat Kernel Pagerank as a Classifier

In [50], Lin and Cohen introduce a simple semi-supervised learning model using personalized pagerank directly as a classifier. In fact, their method consists of simply calculating the personalized pagerank for different $f$, and assigning the class number to each node depending on which distribution gives a higher value of the pagerank for that node. Note that, very often, the exponential sum of heat kernel pagerank converges more rapidly than the geometric sum of personalized pagerank. Therefore, in this section, we describe a procedure based directly on that of [50], but using heat kernel pagerank instead of personalized pagerank. The method is a direct use of heat kernel pagerank as a classifier.

For a data set of $k$ classes, the procedure to cluster the set into $k$ parts is:

- Calculate $\{f_1, ..., f_k\}$ by setting $f_i$ at node $x$ to 1 if $x$ is a labeled node assigned to class $i$. Otherwise, set it to 0.
- Normalize $f_i$ to have the $L^1$ norm of 1.
- Calculate $k$ different heat kernel pageranks $\rho_{t,f_i}$ for $i = 1:k$.
- The class of node $x$ is $argmax_i\{\rho_{t,f_i}(x)\}$.

We will refer to this algorithm as $HKPR$ in Section 5, where we describe the results of this method and compare it to the model proposed in this paper. This simple procedure, which uses heat kernel pagerank directly as a classifier, performs well enough; in most cases, the accuracy is only about $1\% - 2\%$ worse than that of our algorithm.

### 4. Semi-Supervised Heat Kernel Pagerank MBO Algorithm

### 4.1. Motivation

To formulate our semi-supervised classification method, we use a variational approach. Recall the general minimization problem for data classification mentioned in Section 1:

$$\min_u \Big\{ E(u) = R(u) + F(u) \Big\},$$

where $R(u)$ is the regularization functional, $F(u)$ is the fidelity term and $u$ is a function defined on the space of the data set. Since the problem we are solving is classification, in this paper, $u(x)$ represents, in some sense, the class representation of node $x$.

Regarding the $R(u)$ term, the total variation functional

$$TV(u) = \int |\nabla u| dx$$

has been used very successfully in image processing applications as a regularization term [33]. In addition, many papers use total variation to approximate the length of the boundary of the binary segmentation function, often referred to as perimeter.

Let us consider another related functional, called the Ginzburg-Landau (GL) functional, originally proposed to describe physical phenomena such as liquid-gas transitions and superconductivity. It can be formulated as

$$GL(u) = \frac{\epsilon}{2} \int |\nabla u|^2 dx + \frac{1}{\epsilon} \int \mathcal{W}(u) dx,$$

where $\mathcal{W}(u)$ is a double-well potential, such as $\mathcal{W}(u) = \frac{1}{4} u^2 (u-1)^2$, and $\epsilon$ is a positive constant. It has been shown [44] that the Ginzburg-Landau functional converges, as $\epsilon \to 0$, to the perimeter, a quantity that is also often approximated by total variation. The relationship between the two functionals also applies in the graphical framework, and can be formulated in terms of gamma convergence. The result is shown in [4], where is it proven that for any $r$, $TV_w$ is the $\Gamma$-limit, in the sense of gamma convergence, of a sequence of graph-based Ginzburg-Landau (GL)-type functionals. Therefore, one can consider the Ginzburg-Landau functional to be a smooth but arbitarily close approximation to total variation.

The advantage of using the Ginzburg-Landau functional for algorithm derivations lies in the fact that it leads to a simple minimization scheme, while the minimization of total variation results in an unwanted nonlinear curvature term. Moreover, the GL scheme contains the Laplace operator, which allows for the usage of heat kernel pagerank.

Following there ideas, the method in [54] is formulated by considering the Ginzburg-Landau functional as a regularization term. The functional plus fidelity term is minimized using gradient descent, which results in the following equation:

$$\dot{u}(t) = -2\epsilon L u(t) - \frac{1}{\epsilon} \mathcal{W}'(u) - \frac{\partial F}{\partial u}. \tag{4.1}$$

Here, note that the Laplace operator is now replaced by the graph Laplacian, since the graphical framework is used. One possibility to solve (4.1), explored by [54], is to split the dynamics into two steps of first propagating by the heat equation with an extra term (derived from the fidelity term) and then propagating by an equation involving the double-well potential. Another possibility, which we follow in this paper, is to split the dynamics into *three* steps:

1. Propagate by the heat equation:

$$\dot{u}(t) = -Lu(t). \tag{4.2}$$

2. Propagate by the equation involving the fidelity term:

$$\dot{u}(t) = -\frac{\partial F}{\partial u}. \tag{4.3}$$

3. Propagate by the following equation:

$$\dot{u}(t) = -\frac{1}{\epsilon}\mathcal{W}'(u). \tag{4.4}$$

To obtain an approximate solution of (4.1) at discrete times, one would alternate between the three steps. The advantage of this splitting is that step 1 involves solving only the heat equation, and we pursue this splitting to formulate our method. The heat equation is solved using heat kernel pagerank.

Note that, in the limit as $\epsilon \to 0$, the last step becomes thresholding; therefore, it is equivalent to

$$u(x) = \begin{cases} 1 & \text{if } v(x) \geq \frac{1}{2}, \\ 0 & \text{if } v(x) < \frac{1}{2}, \end{cases}$$

where $v$ is the solution to the second step of the dynamics. Due to the binary answer in the last step, this procedure is applicable only to binary problems. Since we would like our algorithm to be applicable to any data set, binary or not, the next section will adapt the procedure to an arbitrary amount of classes.

### 4.2. Algorithm

To formulate the new algorithm, we first define some notation. In the binary case of only two classes, it makes sense to define $u(x)$ for each $x$ to be a real-valued number, representing, in some sense, one of the two classes. However, in the multiclass case, one has to be more careful about defining the label vector to avoid any biases between the classes. Therefore, in the case of $m$ classes and $n$ data points, instead of a vector $u$, we use a matrix $\mathbf{U} = [\mathbf{u}_1; \ldots; \mathbf{u}_n]$, where every node is associated with a probability distribution $\mathbf{u}_i$ (row vector) over the $m$ classes. In other words, the $j^{th}$ entry of $\mathbf{u}_i$ represents the probability node $i$ belongs to class $j$. While $\mathbf{u}_i$ has dimension $1 \times m$, $\mathbf{U}$ has dimension $n \times m$. Moreover, the vector $\mathbf{u}_i$ is an element of the Gibbs simplex $\Sigma^m$, formulated as

$$\Sigma^m := \left\{ (x_1, \ldots, x_m) \in [0,1]^m \,\middle|\, \sum_{i=1}^{m} x_i = 1 \right\}.$$

We denote the vertices of the simplex by $\boldsymbol{e}_i$ for $i = 1:m$; the vector $\boldsymbol{e}_i$ contains all zero values, except the $i^{th}$ entry, which is equal to 1. These vertices correspond to phases whose class designation is known.

Using the standard Gibbs-simplex $\Sigma^m$, it is not hard to extend the theory covered in the previous section to an arbitrary amount of classes. We do not change steps (4.2) and (4.3), except that the label vector will now be a matrix $\mathbf{U}$, where each row is a probability distribition. The third step, however, has to be modified to pertain to any number of classes, so we first project each row of $\mathbf{U}$ to the simplex (using the approach formulated in [16]) and then we displace towards the closest vertex in the Gibbs simplex.

The new algorithm thus consists of the following steps. First, an initial $\mathbf{U}$ is chosen. For fidelity points, we initialize the row of $\mathbf{U}$ to be the corresponding vertex of the simplex. For the rest of the points, we choose the probability distributions randomly. To obtain the next iterate of the matrix $\mathbf{U}$, three steps are taken:

1. Propagation using the heat equation:
$$\dot{\mathbf{U}} = -L\mathbf{U}. \tag{4.5}$$
2. Propagation using:
$$\dot{\mathbf{U}} = -\frac{\partial F}{\partial u}(\mathbf{U}) \tag{4.6}$$
3. Thresholding:
$$\mathbf{u}_i{}^{n+1} = \boldsymbol{e}_h, \tag{4.7}$$
where vertex $\boldsymbol{e}_h$ is the vertex in the simplex closest to the projection of the $i^{th}$ row of the result of step 2 to the simplex, using the procedure in [16].

We alternate between the three steps until there is little difference between the current and the previous iterate, using the metric:
$$\frac{\|\mathbf{u}_i{}^{n+1} - \mathbf{u}_i{}^{n}\|_2^2}{\|\mathbf{u}_i{}^{n+1}\|_2^2} < \eta,$$
where $\eta$ is a small positive constant. The final classification is obtained by assigning to node $i$ the class that holds the highest probability in its class distribution $\mathbf{u}_i$.

Step (4.5) is solved using heat kernel pagerank, which we calculate using two different approaches, to be described in more detail in the next section. To be more specific, first let $c_i$ represent the $i^{th}$ column of the result of step (4.5) and $c_{i0}$ represent the $i^{th}$ column of $\mathbf{U}$ at the start of step (4.5). Now, note that (4.5) consists of solving a system of heat equations in graph form:
$$\begin{cases} \dot{u}(t) = -L_w u(t), & u(0) = c_{00} \in R^n \\ \dot{u}(t) = -L_w u(t), & u(0) = c_{10} \in R^n \\ \quad\vdots & \quad\vdots \\ \dot{u}(t) = -L_w u(t), & u(0) = c_{m0} \in R^n, \end{cases} \tag{4.8}$$
where $m$ is the number of classes. We can use heat kernel pagerank to solve each of those problems; details were given in Section 2.2. More precisely, it is known that column $i$ of the result of step (4.5), which is the solution of the $i^{th}$ equation above, is given by
$$c_i = D^{-1}\rho^{tr}_{t,f}, \qquad f = c_{i0}{}^{tr} D.$$
The $c_i$'s are then concatenated to form the matrix $[c_1 \dots c_m]$, which is the result of step (4.5). Step (4.6) is calculated explicitly, i.e.
$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{dt} = -\frac{\partial F}{\partial u}(\mathbf{U}^n).$$

The heat kernel pagerank MBO algorithm is outlined in Figure 4.3. The version of the algorithm that calculates heat kernel pagerank using series will be referred to as HKPR1; the version that uses random walks to calculate heat kernel pagerank will be referred to as HKPR2.

Fig. 4.1: Heat Kernel Pagerank Approximation (Series)

Heat Kernel Pagerank Approximation (Series)

**Require:** a row vector $f$, probability matrix $P$ of size $n\times n$, $t\in R^+$, maximum $max$ number of terms to include in the series, tolerance $tol$

$\star$ Initialize $s=e^{-t}f$, $r_1=f$, $k=1$, $c=1$.
**while** $m>tol$ and $c<max$ **do**
$\star$ $r_1\leftarrow r_1P$
$\star$ $r_2\leftarrow e^{-t}\frac{t^k}{k!}r_1$
$\star$ $s\leftarrow s+r_2$
$\star$ $m\leftarrow ||r_2||_\infty$
$\star$ $k\leftarrow k+1$, $c\leftarrow c+1$
**end while**
**return** s

Fig. 4.2: Heat Kernel Pagerank Approximation (Random Walk)

Heat Kernel Pagerank Approximation (Random Walk)

**Require:** a row vector $f$, probability matrix $P$ of size $n\times n$, $t\in R^+$, an upper limit $K$ on length of a walk, a number $r$ of random walks

Write $f=f_+-f_-$.
Initialize row vectors $\mu_+$ and $\mu_-$ to zero row vectors of length $n$.
**for** $i=1:r$ **do**
$\star$ Choose a starting vertex $v_0$ using the probability distribution $\frac{f_+}{||f_+||}$.
$\star$ Simulate a random walk starting from $v_0$ using probability matrix $P$, where the number of steps $k$ taken is a Poisson random variable (with parameter $t$), and $k\leq K$.
$\star$ Let $v_f$ be the last vertex of the walk.
$\star$ $\mu_+(v_f)\leftarrow\mu_+(v_f)+1$
**end for**
Repeat the $for$ loop using probability distribution $\frac{f_-}{||f_-||}$ and $\mu_-$.
**return** $\frac{||f_+||}{r}\mu_+ - \frac{||f_-||}{r}\mu_-$

### 4.3. Computing Heat Kernel Pagerank

Step (4.5) of the new algorithm is solved using heat kernel pagerank. Recall the formula for heat kernel pagerank:

$$\rho_{t,f}=fH_t=\sum_{k=0}^{\infty}e^{-t}\frac{t^k}{k!}fP^k. \tag{4.9}$$

We calculate the pagerank via two different approaches: using series and using random walks, to be described in the next two subsections.

#### 4.3.1. Computing Heat Kernel Pagerank Using Series

The first approach is to sum each term of the series (4.9) until the maximum value of the last term is small enough. In other words, each term of the series (4.9) is efficiently calculated until

$$e^{-t}\frac{t^k}{k!}fP^k<\epsilon,$$

Fig. 4.3: Heat Kernel Pagerank MBO Algorithm

Heat Kernel Pagerank MBO Algorithm for Classification

**Require:** a graph $G$ of size $n$, degree matrix $D$, probability matrix $P$, matrix $U$ holding the probability distributions over classes, number of classes $m$, class matrix $C$ ($n \times 1$ matrix) holding the class assignments, $t \in R^+$, tolerance $tol$

Initialize $U$ to $U_0$.
⋆ $U = U_0$, $U_{old} = U_0$.
**while** $d < tol$ **do**
⋆ $U_{old} \leftarrow U$
**for** $i = 1 : m$ **do**
⋆ Let $c_i$ be the $i^{th}$ column of $U$.
⋆ Calculate the heat kernel pagerank $\rho_{t,f}$ using $P$ with the row vector $c_i^{tr} D$, using series for version one (HKPR1) and using random walks for version two (HKPR2).
⋆ $c_i \leftarrow D^{-1} \rho_{t,f}^{tr}$
**end for**
⋆ $U \leftarrow [c_1 \dots c_m]$
⋆ Propagate $U$ using

$$\dot{U} = -\frac{\partial F}{\partial u}(U)$$

⋆ Project each row of $U$ to the simplex using [16].
⋆ Replace each row of $U$ by the closest vertex in the Gibbs simplex
⋆ $d \leftarrow \frac{\|U - U_{old}\|_2^2}{\|U\|_2^2}$
**end while**
**for** $i = 1 : n$ **do**
$C_i = argmax(U_i)$, where $U_i$ is the $i^{th}$ row of $U$.
**end for**
**return** $C$

for some $k$ and $\epsilon$. This shortened sum approximates heat kernel pagerank. The approach is detailed in Figure 4.1. In practice, we calculate at most $j$ terms of the series, where $j$ is a chosen variable; for the experiments, we use up to 12 terms in the series. We also use a sparse matrix $P$, which makes matrix computations much less costly.

### 4.3.2. Computing Heat Kernel Pagerank Using Random Walks

To give the motivation for the second approach, we first assume that $f$ is a probability distribution. In this case, $fP^k$ is the distribution on the vertices after $k$ random walk steps (with probability matrix $P$), starting with a vertex chosen from the probability distribution $f$. Therefore, heat kernel pagerank can be considered as the expected distribution of the random walk, if $k$ steps are taken with probability $\frac{e^{-t}t^k}{k!}$, with the starting vertex chosen from $f$.

This motivates a random walk procedure to calculate heat kernel pagerank. We first calculate $r$ random walks, where each walk proceeds using the probability transition matrix $P$ for exactly $k$ steps, where $k$ is a Poisson random variable, starting with a vertex chosen from $f$. However, since long walks occur with low probability, we limit the random walk length to $K$ steps. Then, as mentioned before, the distribution over the final vertices of the $r$ random walks can approximate heat kernel pagerank.

Since $f$ might not be a probability distribution, we first decompose it into positive and negative portions, $f_+$ and $f_-$, and then normalize each part to sum to one. In other

words, note that

$$\rho_{t,f} = f H_t = (f_+ - f_-)H_t = ||f_+||\Big\{\frac{f_+}{||f_+||}H_t\Big\} - ||f_-||\Big\{\frac{f_-}{||f_-||}H_t\Big\}, \tag{4.10}$$

where, of course, $\frac{f_+}{||f_+||}$ and $\frac{f_-}{||f_-||}$ are probability distributions. The random walk procedure is applied to both of these probability distributions, and the pagerank is obtained using the linear combination in (4.10). The details of this approach are summarized in Figure 4.2. For the experiments, we calculate $r$ and $K$ using the formulas:

$$r = \frac{16}{\epsilon^3} log n, \qquad K = \frac{log(\epsilon^{-1})}{loglog(\epsilon^{-1})},$$

where we use $\epsilon = 0.075$.

## 5. Experimental results

In this section, we validate our method by presenting results for experiments on benchmark data sets. The accuracy of our algorithm is comparable with or better than that of state-of-the-art methods. Even when heat kernel pagerank is used directly as a classifier in a very simple procedure (HKPR method described in Section 3), the performance is still good.

### 5.1. Two Moons

This data set is constructed from two half circles in $\mathbb{R}^2$ with a radius of one. The centers of the two half circles are at (0,0) and (1,0.5). A thousand uniformly chosen points are sampled from each circle, embedded in $\mathbb{R}^{100}$ and i.d.d. Gaussian noise with standard deviation 0.02 is added to each coordinate. Therefore, the set consists of two thousand points. Starting from some initial classification of the points, the goal is to segment the two half circles. The graph is constructed using the 100-dimensional points as feature vectors, and is sparsified based on 10 nearest neighbors with local scaling using the $10^{th}$ closest neighbor. We use fidelity sets of 50 out of 2000 points.

We performed 100 different experiments, each with a different fidelity set. For the first version of the new algorithm (HKPR1), we obtained an average accuracy of 96.26% over 100 runs, with each taking just over a second on average. The average number of iterations was 3.29. For the second version of the new algorithm (HKPR2), we obtained an average accuracy of 96.02% over 100 runs, each taking around 17 seconds. The average number of iterations was around 8.29. When heat kernel pagerank is used directly as a classifier (HKPR algorithm), the average accuracy was around 95.73%.

The results of the proposed method are shown in Table 5.1, where our algorithm (two versions) is written in italics. We also provide those of other related methods for comparison. While a few of the mentioned methods are unsupervised, they all incorporate some sort of prior knowledge, such as a mass constraint, so we view them as comparable to our semi-supervised approach.

### 5.2. MNIST

The MNIST data set [48] is composed of 70,000 $28 \times 28$ images of handwritten digits 0 through 9. Examples of some of the images are shown in Figure 5.1. The task is to correctly assign each image to a digit; this is a 10 class segmentation problem. The feature vector of each node consists of 784 pixel intensity values present in each image of the data set. To construct the weight matrix, we use a graph sparsified using 8 nearest neighbors with local scaling based on the $8^{th}$ closest neighbor. Note that we do not perform any preprocessing, i.e. the graph is constructed from the raw data directly.

For the fidelity term, 250 images per class (2500 images corresponding to 3.6% of the data) are chosen randomly.

First, we tested the full MNIST dataset, consisting of 70000 images of 10 digits, performing many experiments over different fidelity sets. For the first version of the algorithm, we obtained an average accuracy of 97.52% over 100 runs, each taking just 22.25 seconds on average. The average number of iterations was 8.69. For second version, we obtained an average accuracy of 97.48% over 20 different fidelity sets using only around 3.6% of the points. The procedure took around 1.23 hours to run on average. The average number of iterations for this method was 16. Using heat kernel pagerank directly as a classifier (HKPR method) gives an accuracy of around 96.43%.

We also tested a binary subset of MNIST, consisting of only digits 4 and 9, resulting in a data set of 13,782 nodes. This data set was chosen because these digits are hard to distinguish when handwritten. The graph and the fidelity set was constructed in the same way as for the 70K MNIST data set. We performed experiments with different fidelity sets. For HKPR1, we obtained an average accuracy of 98.28% over 100 runs, each taking just around 1 second on average. The average number of iterations was 14.78. For HKPR2, we obtained an average accuracy of around 98.37% over 20 runs. The procedure took around 8.38 minutes to run, taking around 34 iterations on average. The HKPR method's average accuracy was around 94.94%.

The results of both MNIST experiments are shown in Table 5.1, in addition to those of some of the best methods. Most of the algorithms we compare with, such as linear and nonlinear classifiers, boosted stumps, support vector machines, neural and convolution nets, are all supervised learning approaches, using 60,000 of the images for a training set and 10,000 images for a testing set. From the table, one can see that our method performs very competitively with these procedures, and in most cases outperforms them, even with only 3.6% of the nodes as fidelity. Moreover, we note that, unlike the case with some of the algorithms we compare with, no preprocessing or feature extraction was performed on the raw data in our experiments.

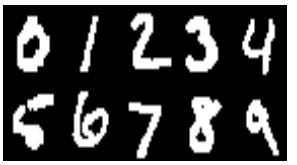

Fig. 5.1: Examples of digits from the MNIST data base

### 5.3. LFR Network

In their paper [46], the authors introduce a class of graphs that are built with heterogeneous distributions of class size and node degree, since many real networks have this property. The degrees and class sizes are assigned from a power-law distribution with power $\xi$ and $\beta$, respectively. The sum of the class sizes must equal the number of nodes $n$ in the graph. In addition, this class of graphs has a mixing parameter $\mu$ associated to them, which is the fraction of edges each node shares with other classes. The maximum degree $d_{max}$, mean degree $d_{av}$, largest class size $s_{max}$, smallest class size $s_{min}$, $\mu$, $\xi$, $\beta$ are parameters that are entered by the user. The code to build these data sets is provided by the authors of [46].

For our experiments, we construct $LFR$ graphs with 1000 nodes, which we call

$LFR1$, and with 50,000 nodes, which we call $LFR50$. To construct the $LFR1$ network, we set the parameters $d_{max}$, $d_{av}$, $s_{max}$, $s_{min}$, $\xi$, $\beta$ to 50, 20, 50, 10, 2, 1, respectively. We vary the $\mu$ parameter from 0.1 to 0.8 in 0.1 increments, since it allows us to control the level of difficulty. A higher $\mu$ would make it more difficult to classify each node. The number of classes in each $LFR1$ network was about 40.

In addition, in order to evaluate our algorithm on larger networks, we construct $LFR50$ networks, each with 50,000 nodes. To construct the network, we first consider 50 LFR1 networks $L_1$,...$L_{50}$, each with a mixing parameter $\mu$, and then connect each node in $L_s$ to $20\mu$ nodes in $L_{s+1}$, chosen uniformly at random, and we note that $L_{51} = L_1$. Since each $LFR1$ network contains around 40 classes, by construction, each $LFR50$ graph consists of around 2000 classes. As in the $LFR1$ case, we vary the mixing parameter $\mu$ of the $LFR1$ networks used to construct $LFR50$, from 0.1 to 0.8 in 0.1 increments. The mixing parameter of the resulting network is approximately $\frac{2\mu}{1+\mu}$. In addition, because of the way the network is constructed, the graph has very similar structure to an $LFR1$ graph. Moreover, the strength of each node in the network is around $1+2\mu$ times that of it in the $LFR1$ network, and the total number of edges in the network is scaled by around $50(1+2\mu)$.

We first evaluate our algorithm on $LFR1$ networks, each with a thousand nodes. For HKPR1, our accuracy depends on the mixing parameter and the size of the fidelity set. The procedure took around 1 second. The results are shown in Table 5.1, where we display the results for mixing parameters 0.1, 0.2, ..., 0.8 and for different sizes of fidelity. The accuracies represent average accuracies obtained over 100 fidelity sets.

We then evaluate the method on the $LFR50$ networks. For HKPR1, our accuracy again depends on the mixing parameter and the size of the fidelity set. The procedures took around 19 minutes. The results are shown in Table 5.1, where we display the results for mixing parameters 0.1, 0.2, ..., 0.8 and for different sizes of fidelity. The accuracies represent average accuracies obtained over 100 different fidelity sets.

**5.4. COIL** We evaluate our performance on the benchmark COIL data set [14,57] from the Columbia University Image Library. This is a set of color $128 \times 128$ images of 100 objects, taken at different angles. The red channel of each image is then downsampled to $16 \times 16$ pixels by averaging over blocks of $8 \times 8$ pixels. Then, 24 of the objects are randomly selected and then partitioned into six classes. Discarding 38 images from each class leaves 250 per class, giving a data set of 1500 data points. To construct the weight matrix, we used 4 nearest neighbors with local scaling based on the $4^{th}$ closest neighbor. The fidelity term is constructed by labeling 10% of the points, selected at random.

We conduct experiments over 100 different fidelity sets. For HKPR1, we obtained an average accuracy of 91.09% over 100 runs, taking just around 1 second on average. The average number of iterations was 8.26. For HKPR2, the average accuracy was around 91.23% over 100 runs. The average number of iterations was around 16, and the computation took around 92 seconds on average. When heat kernel pagerank is used directly as a classifier (HKPR procedure), the average accuracy is 92.12%.

The results of the method are shown in Table 5.1. We also include those of related methods (those that also use 10% of points for fidelity); one can see that our results are very comparable to them or of greater accuracy.

**6. Conclusion**

The beginning of this section will provide a comparison of the proposed method and [31] and show advantages of both. In addition, we present a timing comparison of the algorithms using benchmark data sets in Table 5.2.

Table 5.1: Results for benchmark data sets: Moons, MNIST, LFR and COIL

Two Moons

| Method | Accuracy |
|---|---|
| spectral clustering [30] | 80% |
| p-Laplacian [9] | 94% |
| Cheeger cuts [64] | 95.4% |
| binary GL [3] | 97.7% |
| GL [31] | 98.41% |
| MBO [31] | 98.31% |
| max-flow [51] | 97.10% |
| augmented Lagrangian [51] | 97.07% |
| HKPR (Section 3) | 95.73% |
| *HKPR1 MBO* | 96.26% |
| *HKPR2 MBO* | 96.02% |

MNIST (10 classes)

| Method | Accuracy |
|---|---|
| p-Laplacian [9] | 87.1% |
| multicut normalized 1-cut [39] | 87.64% |
| linear classifiers [47, 48] | 88% |
| Cheeger cuts [64] | 88.2% |
| boosted stumps [42, 48] | 92.3-98.74% |
| transductive classification [65] | 92.6% |
| tree GL [30] | 93.0% |
| $k$-nearest neighbors [47, 48] | 95.0-97.17% |
| neural/convolutional nets [24, 47, 48] | 95.3-99.65% |
| nonlinear classifiers [47, 48] | 96.4-96.7% |
| SVM [27, 47] | 98.6-99.32% |
| GL [31] | 96.8% |
| MBO [31] | 96.91% |
| HKPR (Section 3) | 96.43% |
| *HKPR1 MBO* | 97.52% |
| *HKPR2 MBO* | 97.48% |

COIL

| Method | Accuracy |
|---|---|
| $k$-nearest neighbors [63] | 83.5% |
| LapRLS [2, 63] | 87.8% |
| sGT [41, 63] | 89.9% |
| SQ-Loss-I [63] | 90.9% |
| MP [63] | 91.1% |
| GL [31] | 91.2% |
| MBO [31] | 91.46% |
| HKPR (Section 3) | 92.12% |
| *HKPR1 MBO* | 91.09% |
| *HKPR2 MBO* | 91.23% |

MNIST (2 classes)

| Method | Accuracy |
|---|---|
| GL [31] | 98.37% |
| MBO [31] | 98.29% |
| max-flow [51] | 98.48% |
| augmented Lagrangian [51] | 98.44% |
| HKPR (Section 3) | 94.94% |
| *HKPR1 MBO* | 98.28% |
| *HKPR2 MBO* | 98.37% |

LFR1

| % of points belonging to fidelity / Mixing parameter | 4% | 8% | 12% | 16% | 20% |
|---|---|---|---|---|---|
| 0.1 | 99.38% | 100% | 100% | 100% | 100% |
| 0.2 | 97.62% | 100% | 100% | 100% | 100% |
| 0.3 | 86.20% | 99.08% | 100% | 100% | 100% |
| 0.4 | 64.13% | 99.08% | 99.96% | 100% | 100% |
| 0.5 | 40.50% | 80.70% | 98.36% | 99.54% | 99.95% |
| 0.6 | 29.36% | 58.34% | 77.84% | 85.77% | 91.70% |
| 0.7 | 21.14% | 35.08% | 45.92% | 56.52% | 65.10% |
| 0.8 | 13.6% | 23.21% | 29.80% | 37.64% | 43.00% |

LFR50

| % of points belonging to fidelity / Mixing parameter | 4% | 8% | 12% | 16% | 20% |
|---|---|---|---|---|---|
| 0.1 | 100% | 100% | 100% | 100% | 100% |
| 0.2 | 100% | 100% | 100% | 100% | 100% |
| 0.3 | 99.95% | 99.99% | 99.99% | 99.99% | 99.99% |
| 0.4 | 96.87% | 99.31% | 99.55% | 99.70% | 99.81% |
| 0.5 | 70.38% | 93.57% | 97.83% | 99.10% | 99.37% |
| 0.6 | 39.84% | 60.14% | 75.80% | 81.99% | 89.71% |
| 0.7 | 26.26% | 44.75% | 53.69% | 60.20% | 65.56% |
| 0.8 | 16.68% | 28.14% | 35.58% | 41.76% | 46.63% |

Table 5.2: Timing Comparison (in seconds)

| data set/method | 2 moons | COIL | MNIST (2 digits) | MNIST (10 digits) |
|---|---|---|---|---|
| computation of eigenvectors for MBO [31] | 0.55 | 0.19 | 45.43 | 1683.57 |
| our method's minimization [a] | 1.708 | 1.461 | 1.374 | 24.257 |
| MBO [31] minimization [b] | 2.30 | 1.205 | 0.52 | 15.4 |

[a] This is the timing of the method using already computed weights.
[b] This is the timing of the method using already computed weights and eigenvalues/eigenvectors of the Laplacian.

There are many advantages of the heat kernel pagerank MBO method. First of all, computationally, it is very efficient, as the main part of the algorithm consists of computing heat kernel pagerank, which can be calculated quickly using the series approach or the procedure in [21, 62]. Another advantage of the algorithm to be mentioned is that, unlike in the case of [31], the eigenvalues and the eigenvectors of the graph Laplacian do not need to be calculated. Computing the eigendecomposition might become computationally expensive for large $n$, and this is avoided in our proposed algorithm.

The MBO method [31] has the advantage of providing an efficient approach in the case of *dense* graphs, partly due to the Nyström extension procedure [29] it incorporates. The latter algorithm computes an approximation to the eigendecomposition of the graph Laplacian of a dense graph very quickly using a randomly chosen subset of the nodes. The eigenvectors and eigenvalues are then used to proceed with the MBO method; in fact, the graph weights themselves are not needed for computations after the eigendecomposition has been calculated. Another advantage of [31] is the fact that, even in the case of a very large amount of classes, the procedure is extremely efficient, due to the spectral approach utilized at the main step. In addition, the method is very simple, requiring one to alternate between two uncomplicated steps until a certain stopping criterion is satisfied.

In summary, we have introduced a new efficient semi-supervised algorithm for classification, with results that are competitive with or better than those of state-of-the-art procedures, and accuracy that is high even when the number of labeled points is low. This method, motivated by the approaches of [31], introduces the usage of heat kernel pagerank for its main steps. Overall, this procedure is especially useful for large, sparse graphs and forms a powerful approach to the classification problem.

**Acknowledgements.** We would like to thank Zach Boyd for helpful suggestions and useful discussions.

## REFERENCES


[1] E. Bae and E. Merkurjev. Convex variational methods for multiclass data segmentation on graphs. submitted, 2016.

[2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

[3] A.L. Bertozzi and A. Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Model. Simul.*, 10(3):1090–1118, 2012.

[4] A.L. Bertozzi and Y. van Gennip. Gamma-convergence of graph Ginzburg-Landau functionals. *Adv. Differential Equations*, 17(11–12):1115–1180, 2012.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[6] X. Bresson, T. Laurent, D. Uminsky, and J.H. von Brecht. Convergence and energy landscape for Cheeger cut clustering. *Adv. Neural Inf. Process. Syst.*, 25:1394–1402, 2012.

[7] X. Bresson, X.-C. Tai, T.F. Chan, and A. Szlam. Multi-class transductive learning based on $\ell^1$ relaxations of Cheeger cut and Mumford-Shah-Potts model. *J. Math. Imaging Vision*, 49(1):191–201, 2014.

[8] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Model. Simul.*, 4(2):490–530, 2005.

[9] T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 81–88, 2009.

[10] T.F. Chan, G.H. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *SIAM J. Sci. Comput.*, 20(6):1964–1977, 1999.

[11] T.F. Chan, A. Marquina, and P. Mulet. High-order total variation-based image restoration. *SIAM J. Sci. Comput.*, 22(2):503–516, 2000.

[12] T.F. Chan and C.-K. Wong. Total variation blind deconvolution. *IEEE Trans. Image Process.*, 7(3):370–375, 1998.

[13] H. Chang, X. Zhang, X.-C. Tai, and D. Yang. Domain decomposition methods for nonlocal total variation image restoration. *J. Sci. Comput.*, 60(1):79–100, 2014.

[14] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*, volume 2. MIT Press, 2006.

[15] J. Cheeger. *A lower bound for the smallest eigenvalue of the Laplacian.* 1970.

[16] Y. Chen and X. Ye. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

[17] F. Chung. *Spectral Graph Theory*, volume 92. American Mathematical Society, 1997.

[18] F. Chung. The heat kernel as the pagerank of a graph. *Proc. Nat. Acad. Sci.*, 104(50):19735–19740, 2007.

[19] F. Chung. A local graph partitioning algorithm using heat kernel pagerank. *Internet Math.*, 6(3):315–330, 2009.

[20] F. Chung and O. Simpson. Solving linear systems with boundary conditions using heat kernel pagerank. In *Algorithms and Models for the Web Graph*, pages 203–219. Springer, 2013.

[21] F. Chung and O. Simpson. Computing heat kernel pagerank and a local clustering algorithm. In *Combinatorial Algorithms*, pages 110–121. Springer, 2014.

[22] F. Chung and O. Simpson. Distributed algorithms for finding local clusters using heat kernel pagerank. In *Algorithms and Models for the Web Graph*, pages 177–189. Springer, 2015.

[23] F. Chung and A. Tsiatas. Finding and visualizing graph clusters using pagerank optimization. *Internet Mathematics*, 8(1-2):46–72, 2012.

[24] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1237–1242, 2011.

[25] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7):1384–1399, 2011.

[26] C. Couprie, L. Grady, H. Talbot, and L. Najman. Combinatorial continuous maximum flow. *SIAM J. Imaging Sci.*, 4(3):905–930, 2011.

[27] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Mach. Learn.*, 46(1):161–190, 2002.

[28] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE Trans. Image Process.*, 17(7):1047–1060, 2008.

[29] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2), 2004.

[30] C. Garcia-Cardona, A. Flenner, and A.G. Percus. Multiclass diffuse interface models for semi-supervised learning on graphs. In *Proceedings of the 2th International Conference on Pattern Recognition Applications and Methods*, 2013.

[31] C. Garcia-Cardona, E. Merkurjev, A.L. Bertozzi, A. Flenner, and A. Percus. Fast multiclass segmentation using diffuse interface methods on graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(8):1600–1613, 2014.

[32] T. Gerhart, J. Sunu, L. Lieu, E. Merkurjev, J.-M. Chang, J. Gilles, and A. L. Bertozzi. Detection and tracking of gas plumes in LWIR hyperspectral video sequence data. In *SPIE Conference on Defense Security and Sensing*, 2013.

[33] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Model. Simul.*, 7(3):1005–1028, 2008.

[34] D.F. Gleich. Pagerank beyond the web. *SIAM Review*, 57(3):321–363, 2015.

[35] L. Grady. Multilabel random walker image segmentation using prior models. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 763–770, 2005.

[36] L. Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1768–1783, 2006.

[37] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Proceedings of VIIP*, pages 423–429, 2005.

[38] M. Hein and T. Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA. *Adv. Neural Inf. Process. Syst.*, 23:847–855, 2010.

[39] M. Hein and S. Setzer. Beyond spectral clustering - tight relaxations of balanced graph cuts. *Adv. Neural Inf. Process. Syst.*, pages 2366–2374, 2011.

[40] H. Hu, T. Laurent, M. Porter, and A.L. Bertozzi. A method based on total variation for network modularity optimization using the MBO scheme. *SIAM J. Appl. Math.*, 73(6):2224–2246, 2013.

[41] T. Joachims. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, volume 20, page 290, 2003.

[42] B. Kégl and R. Busa-Fekete. Boosting products of base classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 497–504, 2009.

[43] K. Kloster and D.F. Gleich. Heat kernel based community detection. In *Proceedings of 20th ACM SIGKDD*, pages 1386–1395, 2014.

[44] R.V. Kohn and P. Sternberg. Local minimizers and singular perturbations. *Proc. Roy. Soc. Edinburgh Sect. A*, 111(1-2):69–84, 1989.

[45] D. Koschützki, K.A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. Centrality indices. In *Network analysis*, pages 16–61. Springer, 2005.

[46] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4):046110, 2008.

[47] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of IEEE*, 86(11):2278–2324, 1998.

[48] Y. LeCun and C. Cortes. The MNIST database of handwritten digits.

[49] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1699 –1712, 2008.

[50] F. Lin and W.W. Cohen. Semi-supervised classification of network data using very few labels. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 192–199, 2010.

[51] E. Merkurjev, E. Bae, A.L. Bertozzi, and X.-C. Tai. Global binary optimization on graphs for classification of high-dimensional data. *J. Math Imaging Vis.*, 52(3):414–435, 2015.

[52] E. Merkurjev, A.L. Bertozzi, K. Lerman, and X. Yan. Modified Cheeger and Ratio cut methods using the Ginzburg-Landau functional for classification of high-dimensional data. in review.

[53] E. Merkurjev, C. Garcia-Cardona, A.L. Bertozzi, A. Flenner, and A.G. Percus. Diffuse interface methods for multiclass segmentation of high-dimensional data. *Appl. Math. Lett.*, 33:29–34, 2014.

[54] E. Merkurjev, T. Kostic, and A. L. Bertozzi. An MBO scheme on graphs for segmentation and image processing. *SIAM J. Imaging Sci.*, 6(4):1903–1930, 2013.

[55] E. Merkurjev, J. Sunu, and A.L. Bertozzi. Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video. In *IEEE International Conference on Image Processing*, pages 689–693, 2014.

[56] B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991.

[57] S.A. Nene, S.K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). *Technical Report CUCS-006-96*, 1996.

[58] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[59] P. Perona and L. Zelnik-Manor. Self-tuning spectral clustering. *Adv. Neural Inf. Process. Syst.*,

17:1601–1608, 2004.

[60] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1):259–268, 1992.

[61] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[62] O. Simpson and F. Chung. Finding consensus in multi-agent networks using heat kernel pagerank. in review.

[63] A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *J. Mach. Learn. Res.*, 12:3311–3370, 2011.

[64] A. Szlam and X. Bresson. A total variation-based graph clustering algorithm for Cheeger ratio cuts. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1039–1046, 2010.

[65] A. Szlam, M. Maggioni, and R.R. Coifman. Regularization on graphs with function-adapted diffusion processes. *J. Mach. Learn. Res.*, 9:1711–1739, 2008.

[66] S.A. Tabrizi, A. Shakery, M. Asadpour, M. Abbasi, and M.A. Tavallaie. Personalized pagerank clustering: a graph clustering algorithm based on random walks. *Physica A: Statistical Mechanics and its Applications*, 392(22):5772–5785, 2013.

[67] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[68] J. Wang, T. Jebara, and S.F. Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1144–1151, 2008.

[69] X. Zhang and T.F. Chan. Wavelet inpainting by nonlocal total variation. *Inverse Probl. Imaging*, 4(1):191–210, 2010.

[70] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[71] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Adv. Neural Inf. Process. Syst.*, 16:321–328, 2004.

[72] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. International Conference on Machine Learning, Banff, Canada, 2004.

[73] X. Zhu. Semi-supervised learning literature survey. Technical report 1530, University of Wisconsin-Madison, 2005.